

WHAT IS CLAIMED IS:

1 1. A processor comprising:
2 an out-of-order microinstruction pointer (μ IP) stack in a
3 microcode (μ code) execution core.

1 2. The processor of claim 1 in which the μ IP stack
2 comprises:

3 an entry number field;
4 a microinstruction pointer (μ IP) field;
5 a back pointer field;
6 a retirement indicator field; and
7 a return pointer field.

1 3. The processor of claim 2 in which the μ IP field is
2 14-bits wide.

1 4. The processor of claim 3 in which the μ IP field has
2 a microinstruction pointer (μ IP) pushed by a first
3 microoperation (μ Op) code and used by a second μ Op code.

1 5. The processor of claim 2 in which the back pointer
2 field has a pointer to a next entry in the μ IP stack for a
3 micro-type of service (μ TOS) bit to point to after a μ Op.

1 6. The processor of claim 2 in which the retirement
2 indicator field has an indication of whether an entry has
3 retired.

1 7. The processor of claim 2 in the return pointer field
2 a pointer to a location in a retirement stack to which an
3 entry is copied after being retired.

1 8. A method executed in a processor comprising:

2 executing microcode (μ code) stored in an out-of-
3 order microinstruction pointer (μ IP) stack; and
4 manipulating the μ IP stack with a set of
5 microinstructions.

1 9. The method of claim 8 in which the stack has an
2 entry number field, a microinstruction pointer (μ IP) field, a
3 back pointer field, a retirement indicator field and a return
4 pointer field.

1 10. The method of claim 9 in which the μ IP pointer field
2 is 14-bits wide.

1 11. The method of claim 10 in which the μ IP pointer
2 field has a microinstruction pointer (μ IP) pushed by a first
3 microoperation (μ Op) code and used by a second μ Op code.

1 12. The method of claim 9 in which the back pointer
2 field has a pointer to a next entry in the μ IP stack for a
3 micro-type of service (μ TOS) bit to point to after a μ Op.

1 13. The method of claim 9 in which the retirement
2 indicator field has an indication of whether an entry has
3 retired.

1 14. The method of claim 9 in which the return pointer
2 field contains a pointer to a location in a retirement stack
3 to which an entry is copied after being retired.

1 15. The method of claim 9 in which manipulating
2 comprises:

3 pushing a next μ IP on to the μ IP stack; and
4 using the next μ IP in an intermediate field as a target
5 μ IP in a jump operation.

1 16. The method of claim 9 in which manipulating
2 comprises:

3 taking a value of an intermediate field of a
4 microoperation (μ Op); and
5 pushing the value on to the μ IP stack.

1

1 17. The method of claim 9 in which manipulating
2 comprises:

3 popping a value off the μ IP stack; and
4 replacing a current μ Op intermediate field.

1 18. The method of claim 9 in which manipulating
2 comprises:

3 popping a value off of the μ IP stack; and
4 jumping to that value.

1 19. The method of claim 9 in which manipulating
2 comprises:

3 reading a value off the μ IP stack; and
4 replacing a μ Op's intermediate field with the value.

1 20. The method of claim 9 in which manipulating
2 comprises setting the μ IP stack pointers to reset.

1 21. The method of claim 9 further comprising providing a
2 set of pointers that point to different entries in the μ IP
3 stack.

1 22. The method of claim 21 in which the set of pointers
2 includes a μ TOS pointer that points to a top of the μ IP stack.

1 23. The method of claim 21 in which the set of pointers
2 includes a μ Alloc pointer that points to a next allocated
3 entry in the μ IP stack.

1 24. The method of claim 21 in which the set of pointers
2 includes a NextRet pointer that points to a next entry in the
3 μ IP stack to be deallocated.

1 25. The method of claim 21 in which the set of pointers
2 includes μ RetTos pointer that points at a retired top of the
3 μ IP stack.

1 26. The method of claim 8 in which the μ OPs include an
2 ms_call μ OP that takes a next μ IP, pushes the next μ IP on the
3 μ IP stack, and uses the next μ IP in an intermediate field as a
4 target μ IP of a jump.

1 27. The method of claim 8 in which the μ OPs include an
2 ms_push μ OP that takes a value in an intermediate field and
3 pushes the value on the μ IP stack.

1 28. The method of claim 8 in which the μ OPs include an
2 ms_pop μ OP that pops a value off the μ IP stack and replaces
3 the value with the μ OP's intermediate field.

1 29. The method of claim 8 in which the μ OPs include an
2 ms_return μ OP that pops a value off of the μ IP stack and jumps
3 to that μ IP.

1 30. The method of claim 8 in which the μ OPs include an
2 ms_tos_read μ OP that reads a value off the μ IP stack and
3 replaces this μ OP's intermediate field.

1 31. The method of claim 8 in which the μ OPs include an
2 ms_mip_stack_clear μ OP that sets the μ IP stack pointers to
3 reset.

1 32. A computer program product residing on a computer
2 readable medium having instructions stored thereon which, when
3 executed by the processor, cause the processor to:
4 execute microcode (μ code) stored in an out-of-order
5 microinstruction pointer (μ IP) stack; and
6 manipulate the μ IP stack with a set of microinstructions.

1 33. The computer program product of claim 32 wherein
2 instructions to manipulate further comprise instructions to:
3 push a next μ IP on to the μ IP stack; and
4 use the next μ IP in an intermediate field as a target μ IP
5 in a jump operation.

1 34. The computer program product of claim 32 wherein
2 instructions to manipulate further comprise instructions to:
3 take a value of an intermediate field of a microoperation
4 (μOp); and
 push the value on to the μIP stack.

1 35. The computer program product of claim 32 wherein
2 instructions to manipulate further comprise instructions to:
3 pop a value off the μIP stack; and
4 replace a current μOp intermediate field with the value.

1 36. The computer program product of claim 32 wherein
2 instructions to manipulate further comprise instructions to:
3 pop a value off of the μIP stack; and
4 jump to that value.

1 37. The computer program product of claim 32 wherein
2 instructions to manipulate further comprise instructions to:
3 read a value off the μIP stack; and
4 replace a μOp's intermediate field with the value.

1 38. The computer program product of claim 32 wherein
2 instructions to manipulate further comprise instructions to:
3 set the μIP stack pointers to reset.